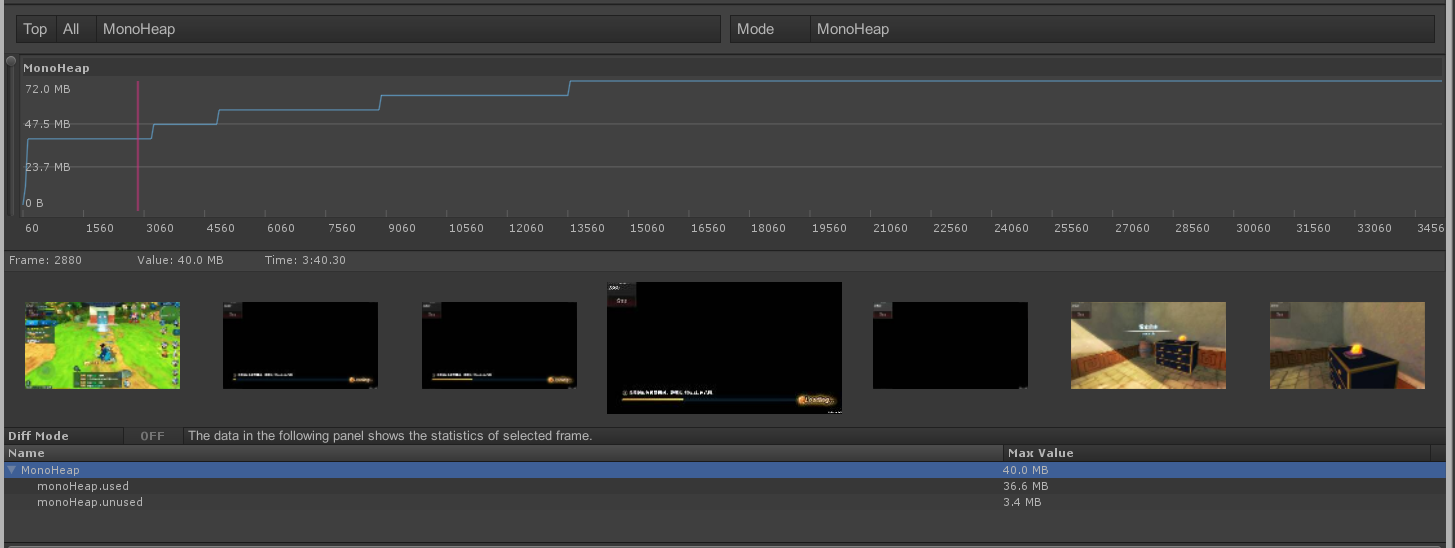
使用UWA 的GOT工具进行mono内存泄漏排查：



使用OverView模式进行数据收集，如上图所示。Monoheap的内存分配情况，其走势与腾讯wetest走势类似。如上图所示在每次mono堆内存上涨都是切换地图造成的。

如下图所示：再切换场景之前已使用的mono堆内存为37.6M（图1），未用的为2.3M（图2），如果在切换场景过程中功能需要申请的内存超过2.3M。mono就会向系统请求新内存。图3表示红框中的方法分配堆内存的驻留情况4.9M（小的时候2M左右）。所以在切换场景时我们的mono内存不够用，需要向系统重新请求。

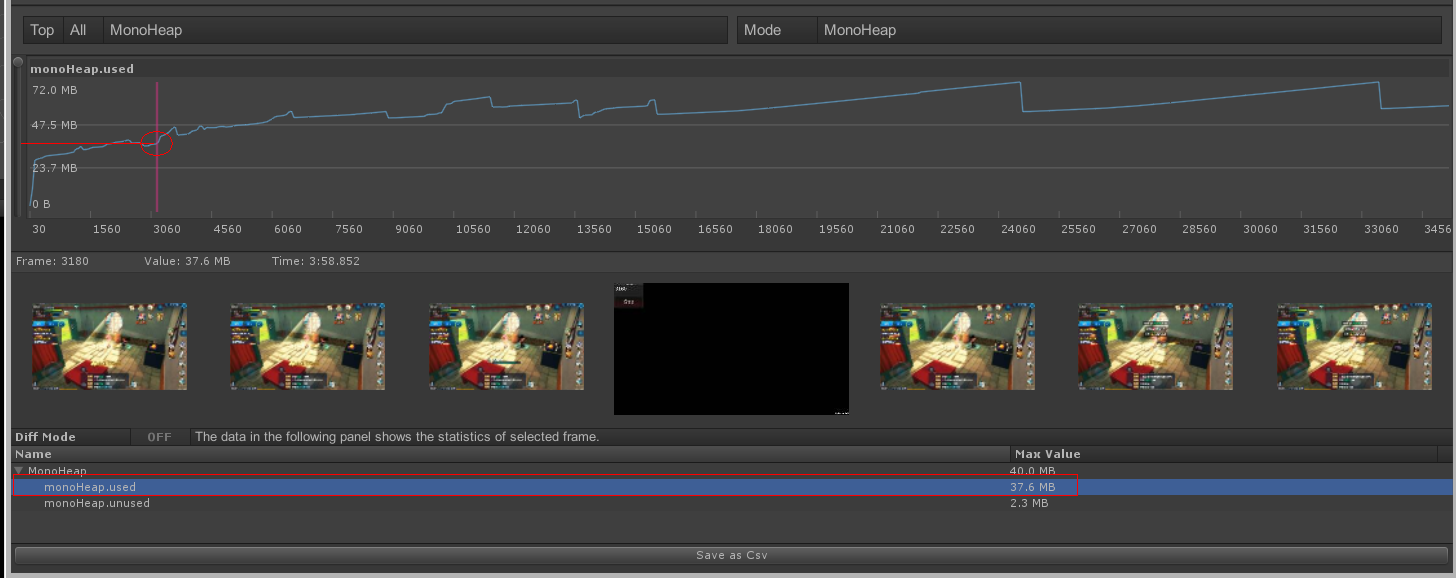


图1

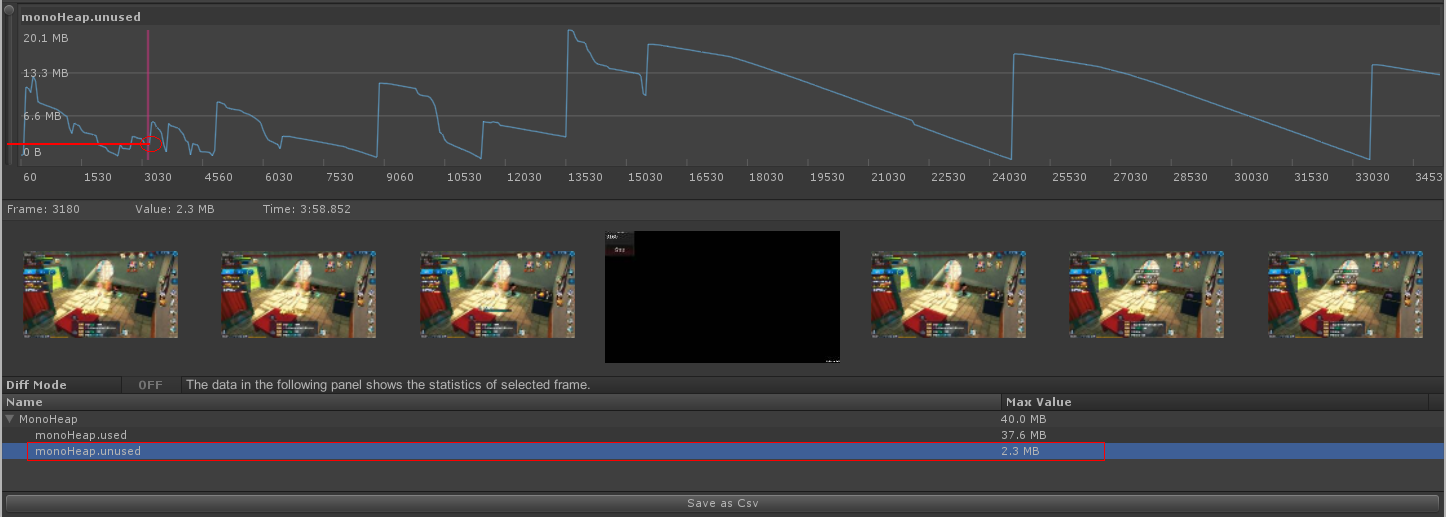


图2

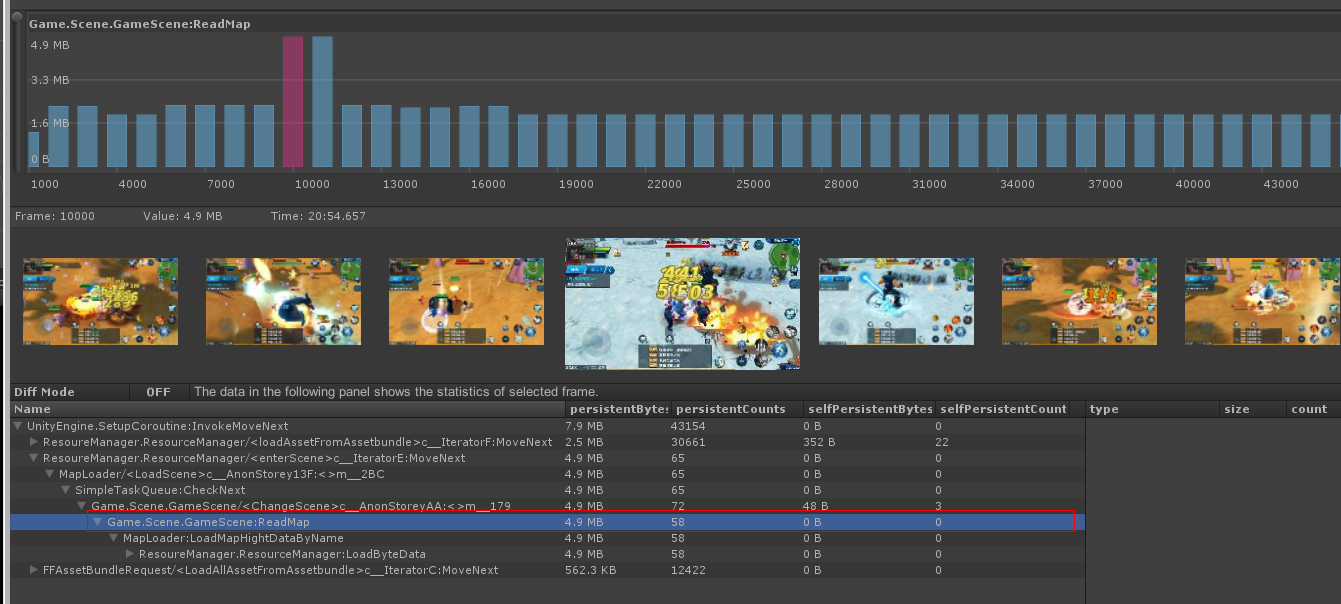
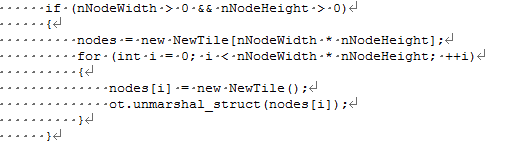


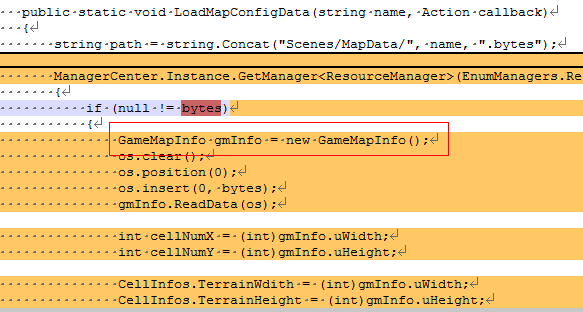
图3

切场景过程的可以简化的new操作：

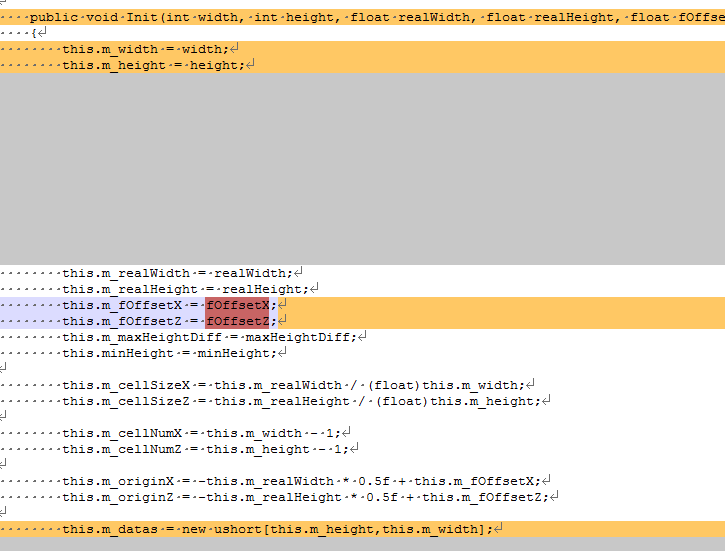
1、GameMapinfo的ReadData方法每次都会new NewTitle的数组，这个数组可以可以重复利用，不用每次都去new，在长度不够时再调用resize方法。



2、MapLoader，中每次调用LoadMapConfigData方法时都会new一个GameMapInfo的对象，这个对像包含了1.中的NewTitle数组，所以每次又会重新new NewTitle数组，会申请大量内存。根据逻辑可以看出，GameMapInfo只是一个数据中转，并没有对象或者方法需要保留它。所以MapLoader中可只缓存一个GameMapInfo的对象，用来数据中转，避免每次切换地图时都做new操作。



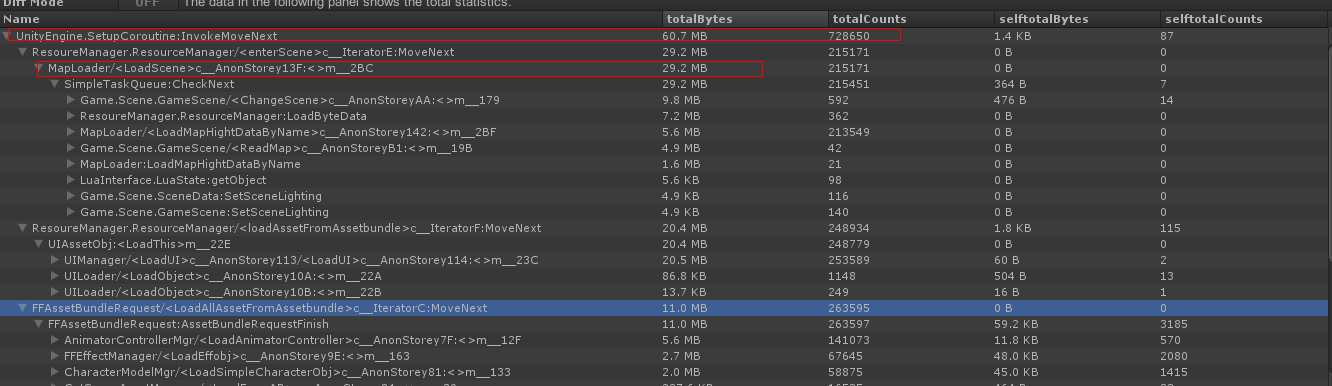
3、初始化高度图，在LoadMapHightDataByName方法中有一个new数组操作，在MapHightData的Init中还有一个new数组操作。优化：去掉中间转化new的数组，直接使用BinaryReader去刷新MapHightData数据。

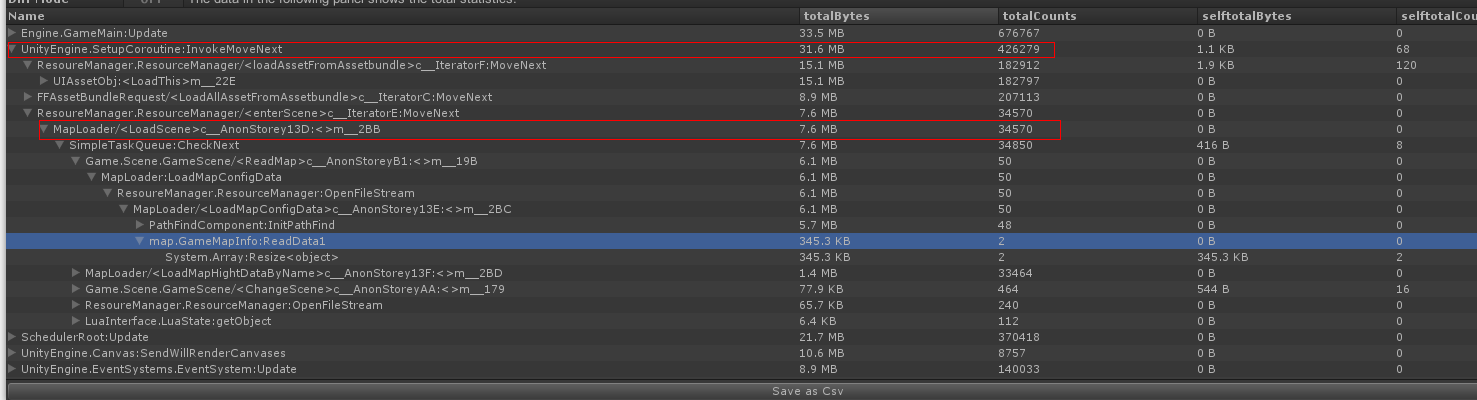


优化前和优化后申请内存情况对比：

优化前：InvokeMoveNext申请内存总大小为60M，LoadScene方法总大小为29.2M

优化后：InvokeMoveNext申请内存总大小为31.6M，LoadScene方法总大小为7.6M



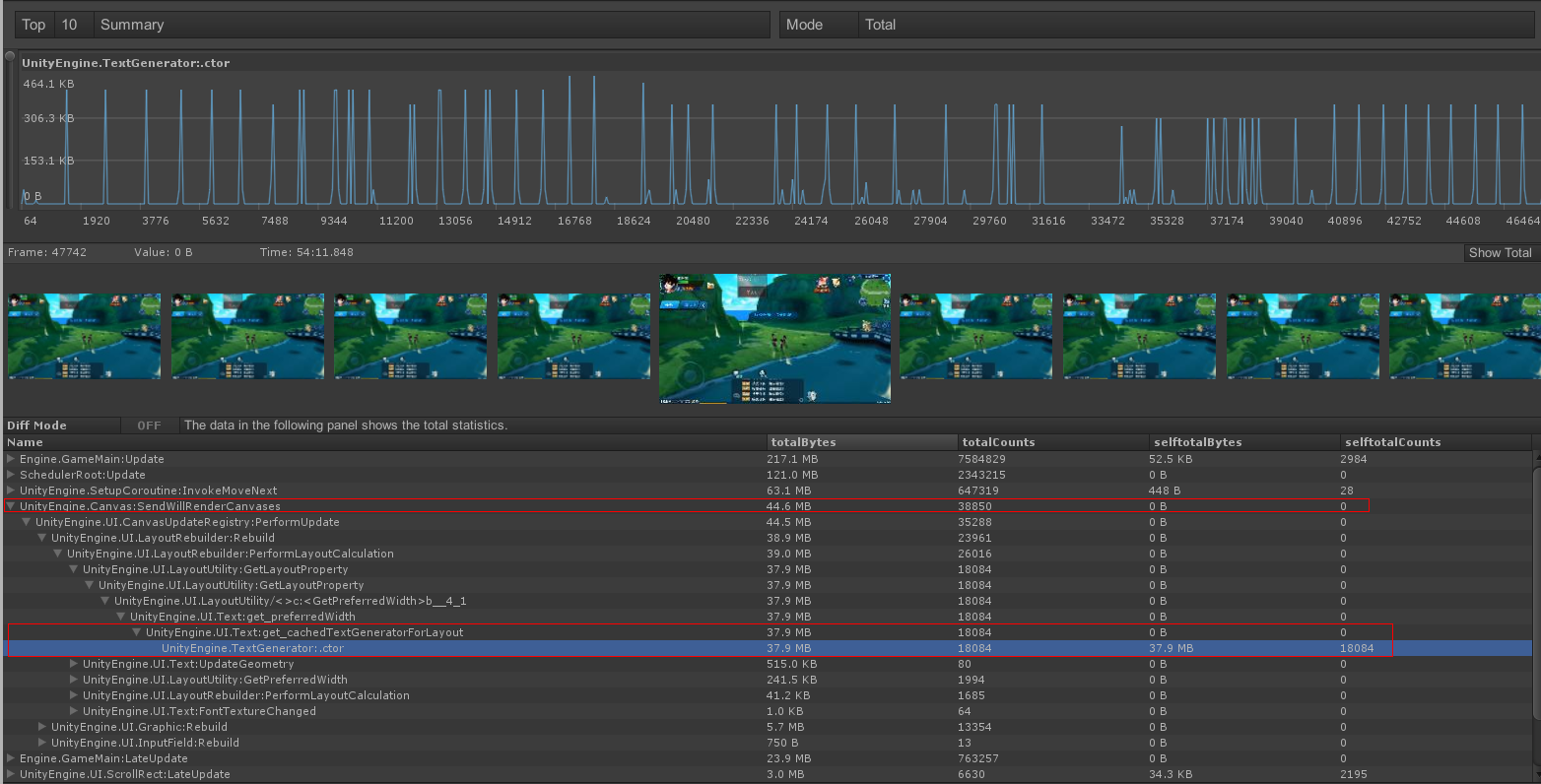


注意：

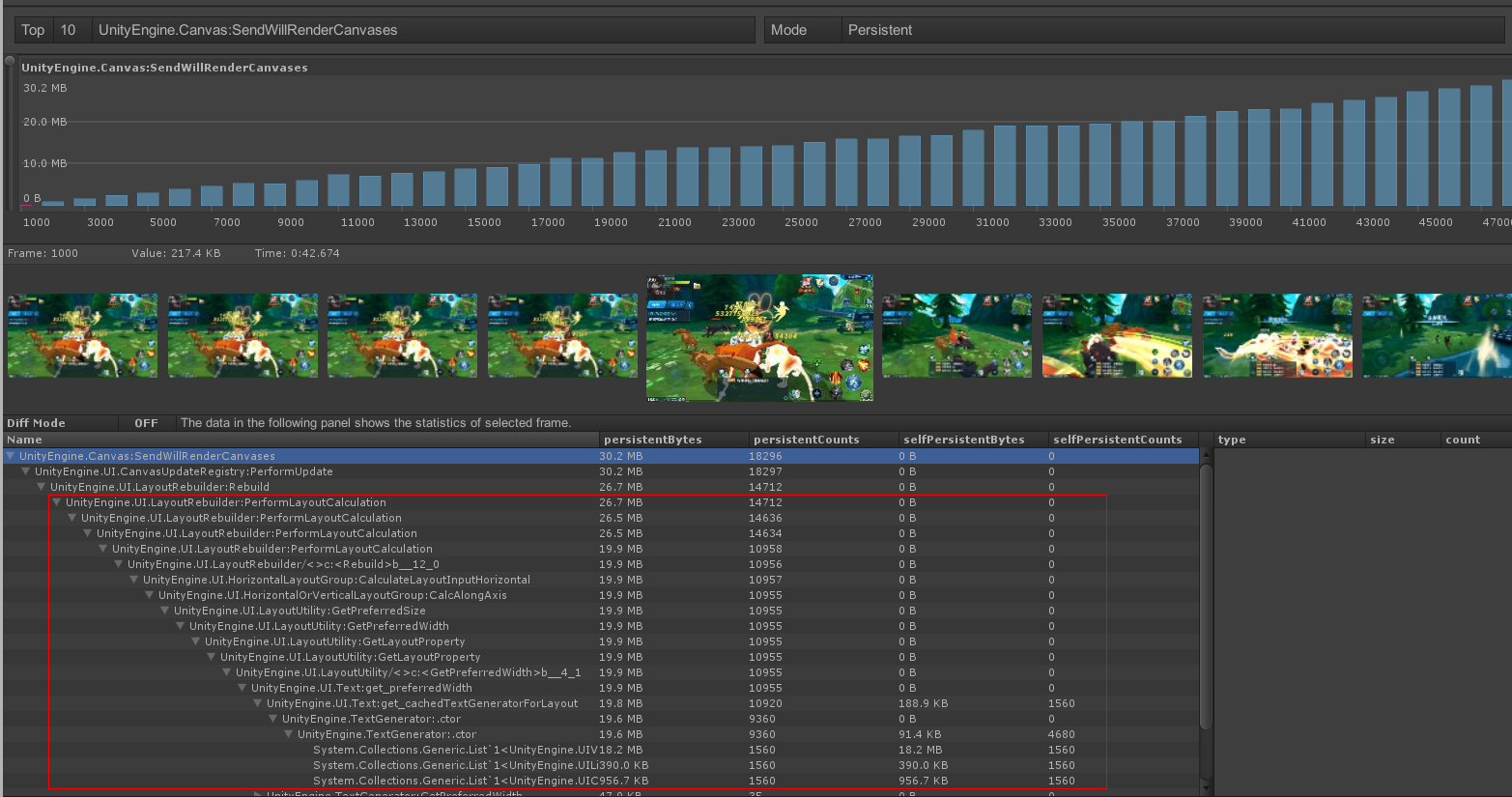
1. 频繁使用的方法（Update，切地图等）中不要大量的做new操作。如果有必要可以把需要频繁使用的对象，数组等缓存下来，下次使用时做refresh操作。
2. 自己系统管理的变量，要管理好它的生命周期，尽量不要有多重引用，让不用的内存及时GC掉。
3. 静态类对象及时卸载
4. 分配内存尽量小，避免申请大块内存，一旦多个大块内存同时存在并且无法释放，则会导致内存过高应用崩溃。

## 2017.11.10

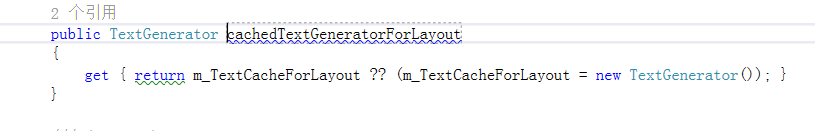
## 使用UWA的GOT工具进行mono内存泄漏排查：



如上图：在Android设备收集数据，Cancas.SendWillRenderCanvases()方法总共申请了44.6M内存。而该方法在申请的mono内存驻留情况如下图所示随时间递增。最后已经达到30.2M。



分析原因：从图中可以看出mono内存的占用是由LayoutRebuilder产生的。从UGUI源码查看结果：在LayoutGroup 和 LayoutElement 两个脚本对list进行排版时，会调用Text类的preferredWidth来获取宽度或者高度是通过TextGenerator 类去获取的，而每一个新的Text都会new一个TextGenerator对象，这个对象是占用mono的主要原因。



## 内存泄漏的原因：

1、主界面的任务和聊天列表

主界面的任务列表刷新是删除所有元素在重新clone新的元素（图4）。聊天界面则是先clone一个元素，在超过范围之后再删除（图5）。每次新clone的元素导致每次新产生text脚本，排版时都会重新new 一个TextGenerator，而lua的Getcompment(“Text”)则会使这个TextGenerator占用的内存无法被释放。

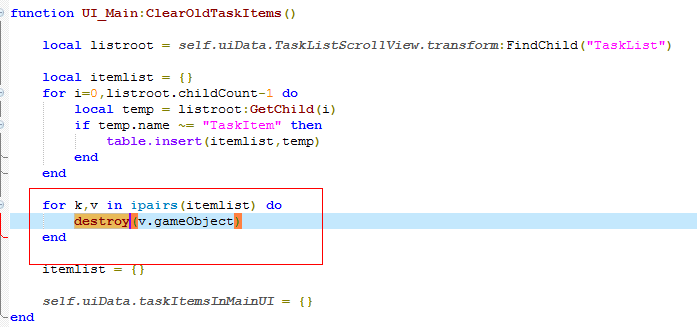
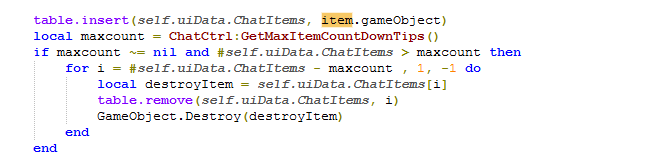
 

图4 图5

2、Lua的GetComponent

Lua中获取Text组件通过GameobjectWrap的GetComponent方法调用，这个方法会把获取的text的object的引用添加到ObjectTranslator类的objects的字典里（图6图7）。而当text的父节点的Gameobject被删除时，text的引用并不会从objects的字典里面移除。导致Text组件中new 的TextGenerator无法被释放（GC掉）知道lua的GC方法被调用，我们的lua的GC只有在切换场景的时候才会调用。

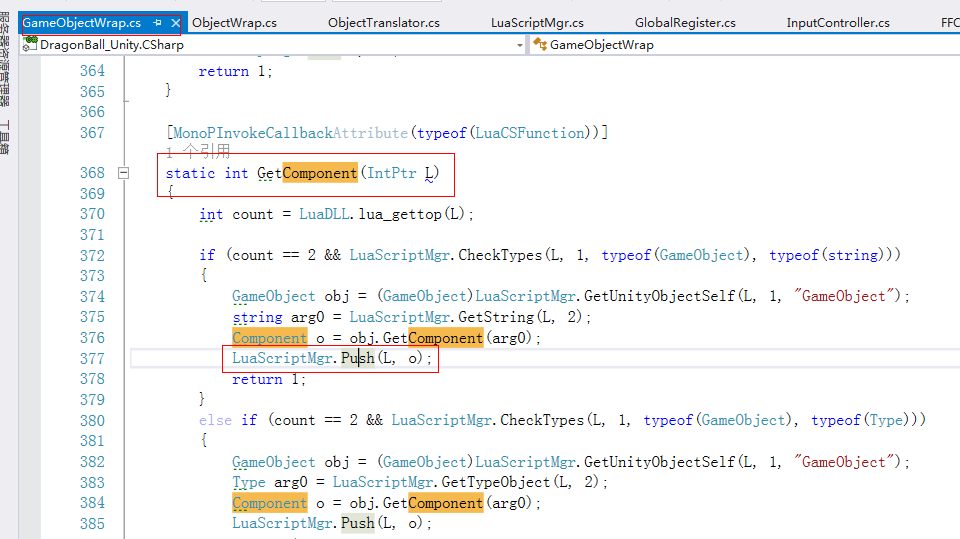
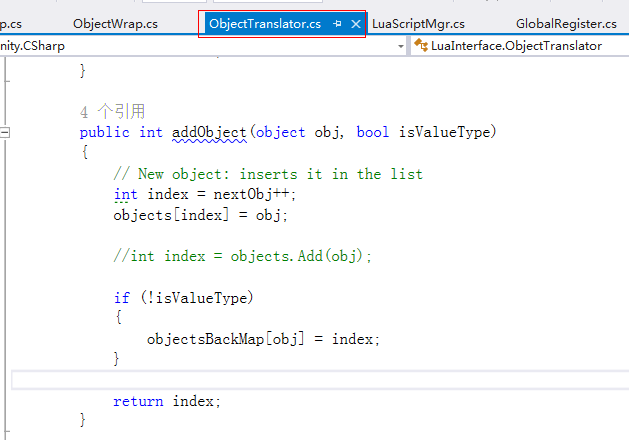
。 

图6 图7

## 解决办法：

1. 把聊天和任务列表都是用可复用的列表。
2. Lua中调用的Getcompment方法通过GR来返回对应的Component。

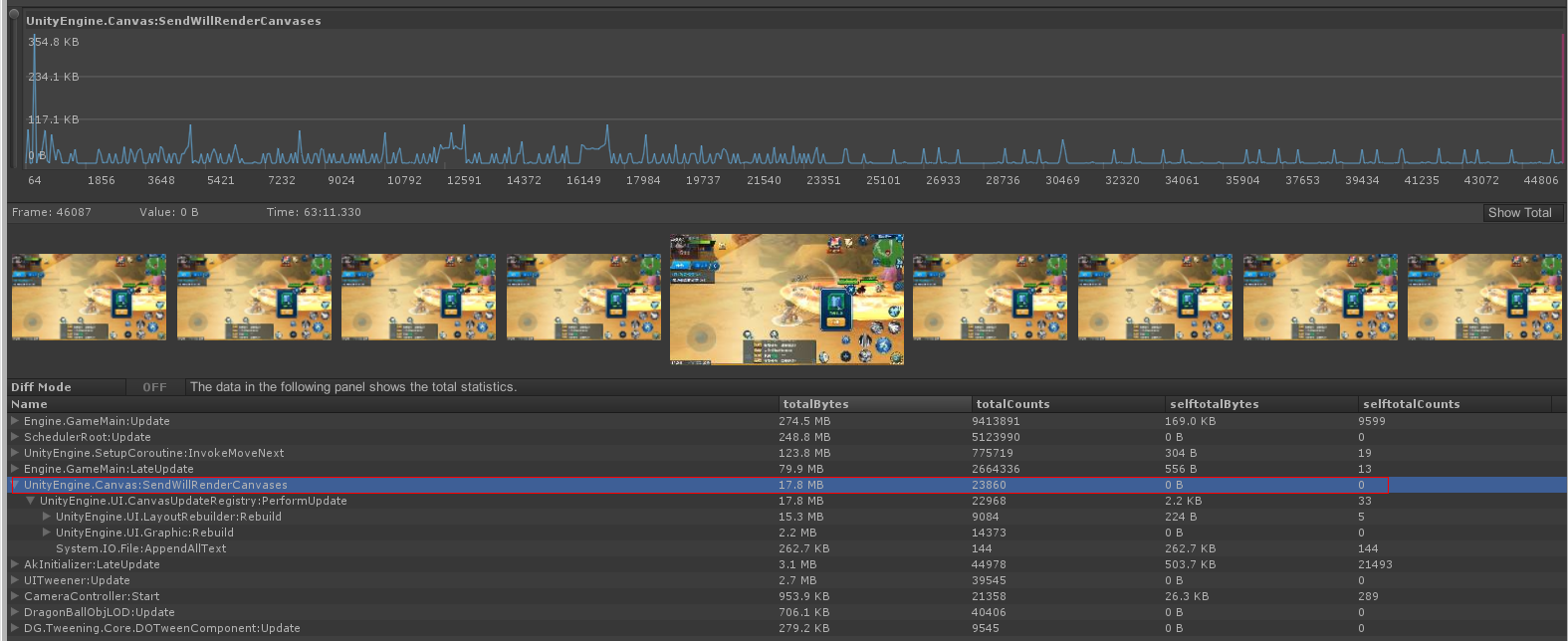
主UI的任务列表和聊天列表都是用可复用的列表后的数据收集。

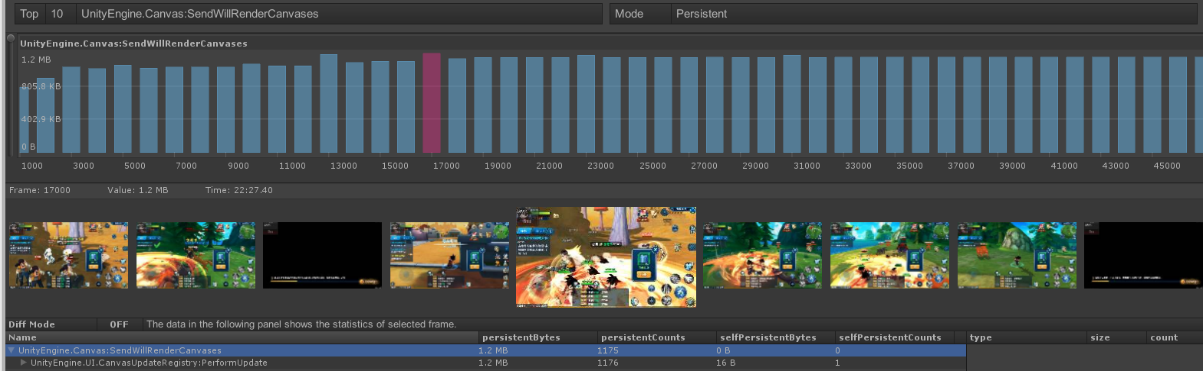
1. 在删除Text的父物体之前，把所有Text的引用remove掉。

## 优化后的数据收集：

1、任务和聊天列表改成可复用列表后，选择mono模式收集数据

Cancas.SendWillRenderCanvases()总共申请了17.8M内存，而在每10000的内存快照中查看，该方法申请的内存驻留情况不再递增，维持在1.1m和1.2m左右。





2、在删除Text的父物体之前，把所有Text的引用remove掉。并且任务和聊天列表改成可复用列表

用Overview查看mono内存情况，50分钟左右的数据收集情况，mono维持在55.3M以下没有再增长。（后续再增加测试时长，看是否能够维持在55.3M以下）

